

Fault Tolerance & PetaScale Systems: Current Knowledge, Challenges and Opportunities*

*preparation of Europar'08 Keynote

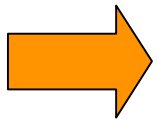
Franck Cappello
INRIA

Second French-Japanese Workshop
Petascale Applications, Algorithms and Programming (PAAP)
Toulouse, June 2008

Challenges of PetaScale computing and beyond

- Application scalability
- Programming models
- Programming env. and debuggers
- Numerical & communication libraries
- Data storage and archive

All these topics are related to Fault Tolerance



Fault tolerance should be considered as a major issue and integrated early in the design of a PetaScale system

Some Issues related to fault tolerance of parallel applications

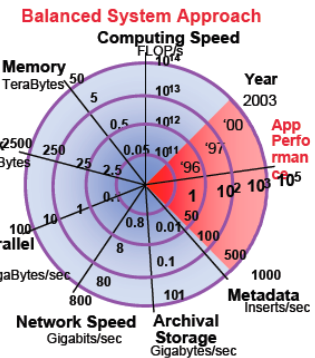
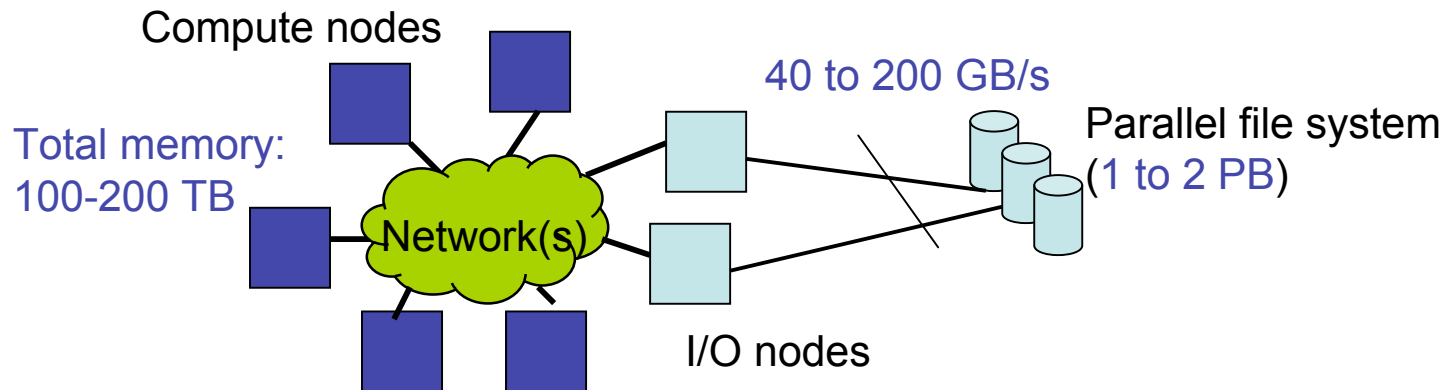
- Why Fault Tolerance is Challenging?
- What are the main reasons behind failures?
- What is the general trend about failure rates?
- Rollback recovery or Replication?
- Rollback recovery Protocol?
- Optimization of Rollback Recovery?
- Alternatives to Rollback recovery?
- Where are the opportunities?

Why Fault Tolerance in large scale systems is challenging

- Difficult (scalability), also for the programmer
 - Not always considered as first class issue
 - Has to live with existing (old) software
 - Has to work on large variety of hardware
 - New results need strong efforts (old research discipline)
 - Etc.
-
- Very expensive, why? --> let's see...

Classical approach for FT: Checkpoint-Restart

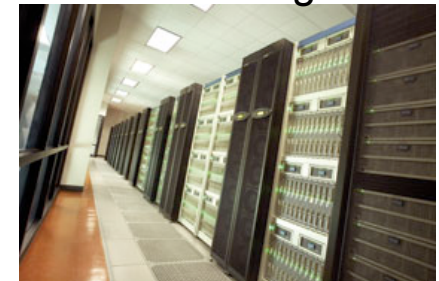
Typical “Balanced System” for PetaScale Computers



RoadRunner



TACC Ranger



LLNL BG/L

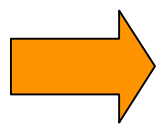


Systems	Perf.	Ckpt time	Source
RoadRunner	1PF	~20 min.	Panasas
LLNL BG/L	500 TF	20 min.	LLNL
Argonne BG/P	500 TF	30 min.	LLNL
TACC Ranger	500 TF	20 min.	LLNL
LLNL Zeus	500 TF	20 min.	LLNL

Checkpoint-Restart needs about 1h!

Cost of Checkpointing

- Use all resources: CPUs (if compression), Memory, Network and Disc --> may be one of the most consuming actions in HPC! (this class of machines consume >1 MegaWatt/h!)
- Slowdown the execution: even if checkpoint is non-blocking, computations and communications are slowed down significantly
- Synchronization & message flush: without specific hardware, sync may take 1 minute or more --> for a 1 Petaflops machine that's equivalent to a cluster of 730 CPUs for 1 day. If you sync 10x per day --> you loose a cluster of 7300 CPUs!
- 1h of a PetaScale systems costs \$2K-\$4K --> How many checkpoints can you afford per day? (estimation based on a cost of \$50K-\$100K per day for a PetaScale machine)



Challenge: Reduce the cost of Checkpointing or imagine novel FT techniques

Some Issues related to fault tolerance of parallel applications

- Why Fault Tolerance is Challenging?
- What are the main reasons behind failures?
- What is the general trend about failure rates?
- Rollback recovery or Replication?
- Rollback recovery Protocol?
- Optimization of Rollback Recovery?
- Alternatives to Rollback recovery?
- Where are the opportunities?

Classical Understanding Approach: Failure logs

- The computer failure data repository (CFDR)

→ <http://cfdrr.usenix.org/>

→ From 96 until now...

→ HPC systems+Google


- failure logs from LANL, NERSC, PNNL, ask.com, SNL, LLNL, etc.

ex: LANL released root cause logs for:
23000 events causing apps. Stop on 22
Clusters (5000 nodes), over 9 years

(I) High-level system information				(II) Information per node category			
HW	ID	Nodes	Procs	Procs /node	Production Time	Mem (GB)	NICs
A	1	1	8	8	N/A – 12/99	16	0
B	2	1	32	32	N/A – 12/03	8	1
C	3	1	4	4	N/A – 04/03	1	0
D	4	164	328	2	04/01 – now	1	1
	5	256	1024	4	12/02 – now	1	1
	6	128	512	4	12/01 – now	16	2
	7	1024	4096	4	09/01 – 01/02	16	2
				4	05/02 – now	8	2
				4	05/02 – now	16	2
				4	05/02 – now	32	2
				4	05/02 – now	352	2
	8	1024	4096	4	10/02 – now	8	2
				4	10/02 – now	16	2
				4	10/02 – now	32	2
				4	10/02 – now	32	2
	9	128	512	4	09/03 – now	4	1
	10	128	512	4	09/03 – now	4	1
	11	128	512	4	09/03 – now	4	1
	12	32	128	4	09/03 – now	4	1
				4	09/03 – now	16	1
				2	09/03 – now	4	1
				2	09/03 – now	4	1
				2	09/03 – now	4	1
				2	09/03 – now	4	1
				2	09/03 – now	4	1
				2	03/05 – 06/05	4	1
	18	512	1024	2	12/96 – 09/02	32	4

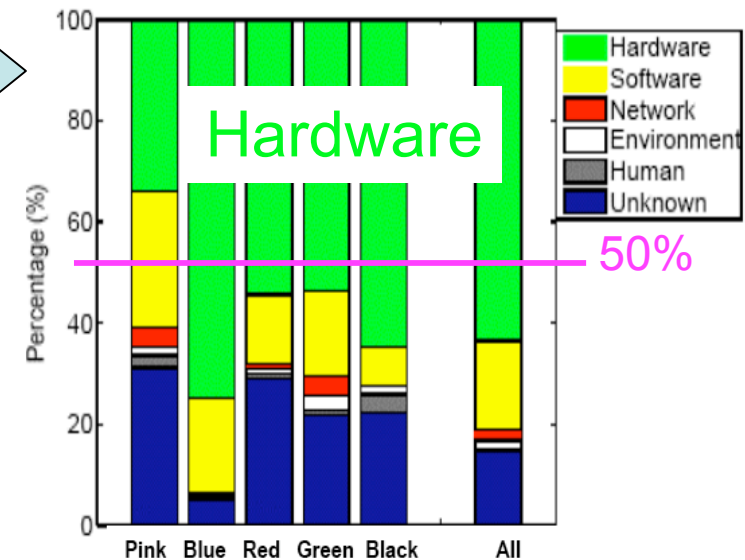
System	Owner	Vendor	Top500 Rank	Procs	Memory (GB)	Interconnect
Blue Gene/L	LLNL	IBM	1	131072	32768	Custom
Thunderbird	SNL	Dell	6	9024	27072	Infiniband
Red Storm	SNL	Cray	9	10880	32640	Custom
Spirit (ICC2)	SNL	HP	202	1028	1024	GigEthernet
Liberty	SNL	HP	445	512	944	Myrinet

Analysis of failure logs

- In 2005 (Ph. D. of CHARNG-DA LU) : “**Software** halts account for the most number of outages (59-84 percent), and take the shortest time to repair (0.6-1.5 hours). Hardware problems, albeit rarer, need 6.3-100.7 hours on the average to solve.”
- In 2007 (Garth Gibson, ICPP Keynote): 

- In 2008 (Oliner and J. Stearley, DSN Conf.):

Type	Raw		Filtered	
	Count	%	Count	%
Hardware	174 586 516	98.04	1 999	18.78
Software	144,899	0.08	6,814	64.01
Indeterminate	3,350,044	1.88	1,832	17.21



Relative frequency of root cause by system type.

Conclusion1: Both Hardware and Software failures have to be considered

Conclusion2: Oliner: logging tools fail too, some key info is missing, better filtering (correlation)

 **Challenge: better logging and analyzing tools**

Some Issues related to fault tolerance of parallel applications

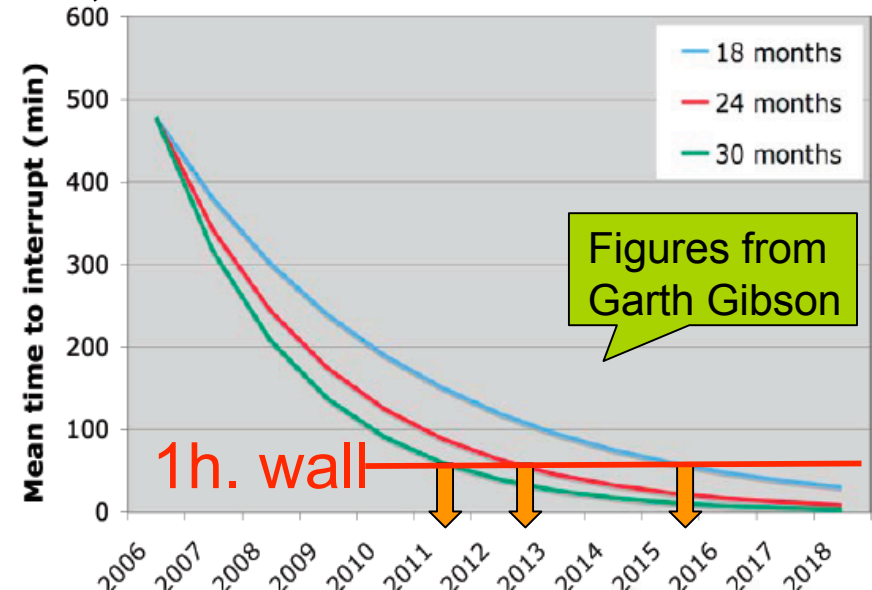
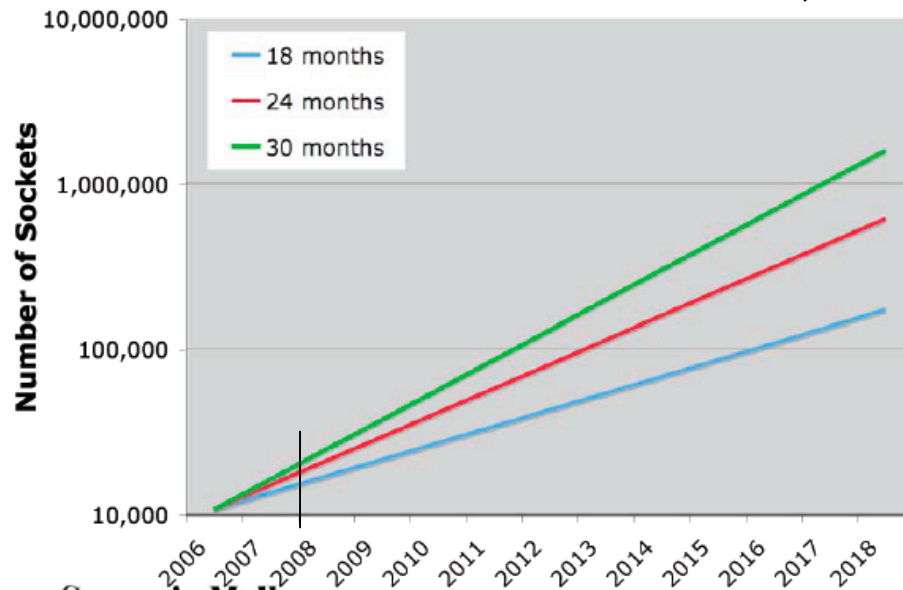
- Why Fault Tolerance is Challenging?
- What are the main reasons behind failures?
- What is the general trend about failure rates?
- Rollback recovery or Replication?
- Rollback recovery Protocol?
- Optimization of Rollback Recovery?
- Alternatives to Rollback recovery?
- Where are the opportunities?

Failure rate and #sockets

In Top500 machine performance X2 per year

--> more than Moore's law and increase of #cores in CPUs

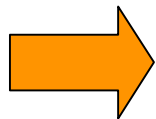
If we consider #core X 2, every 18, 24 and 30 months:



Sockets in increasing rapidly (100K in 2012)

The MTBF of CPU is not likely to improve in the near future

We will reach the 1h. wall as soon as in 2011-2013



Challenge: Improve ckpt time or find alternative approaches

Some Issues related to fault tolerance of parallel applications

- Why Fault Tolerance is Challenging?
- What are the main reasons behind failures?
- What is the general trend about failure rates?
- **Rollback recovery, Replication, Migration?**
- Rollback recovery Protocol?
- Optimization of Rollback Recovery?
- Alternatives to Rollback recovery?
- Where are the opportunities?

Does Replication make sense?

- Classic reliable computing: process-pairs

- Distributed, parallel simulation as transaction (message) processing
- Automation possible w/ hypervisors
- Deliver all incoming messages to both
- Match outgoing messages from both
- 50% hardware overhead + slowdown of pair synch
- No stopping to checkpoint
 - Less pressure on storage bandwidth except for visualization checkpoints

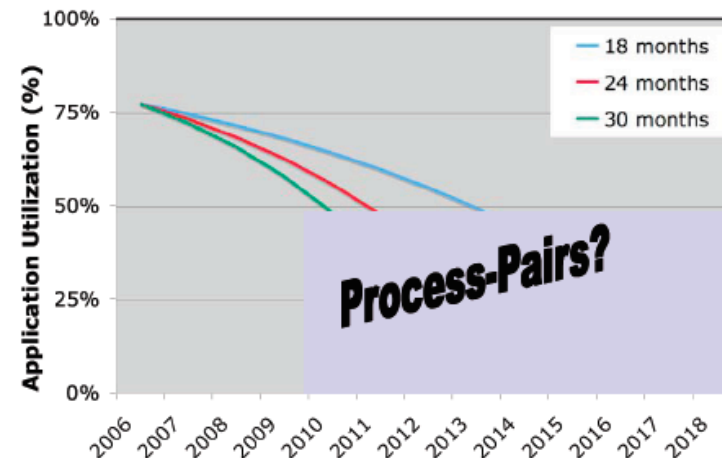
A NonStop* Kernel

Joel F. Bartlett
Tandem Computers Inc.

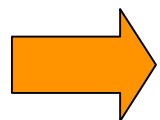
Abstract @ 1981 ACM 0-89791-062-1-12/81-002

The Tandem NonStop System is a fault-tolerant [1], expandable, and distributed computer system designed expressly for online transaction processing. This paper describes the key primitives of the kernel of the operating system. The first section describes the basic hardware building blocks and introduces their software analogs: processes and messages. Using these primitives, a mechanism that allows fault-tolerant resource access, the process-pair, is described. The paper concludes with some observations on this type of system structure and on actual use of the system.

Slide from
Garth Gibson



- Replication (process pairs) can tolerate only 1 simultaneous failure
- Full replication is probably too expensive (double the Hardware AND power consumption)



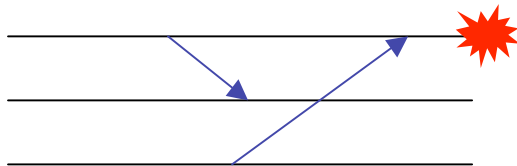
Challenge: Replication (N faults) only for nodes that are likely to fail during the execution

Some Issues related to fault tolerance of parallel applications

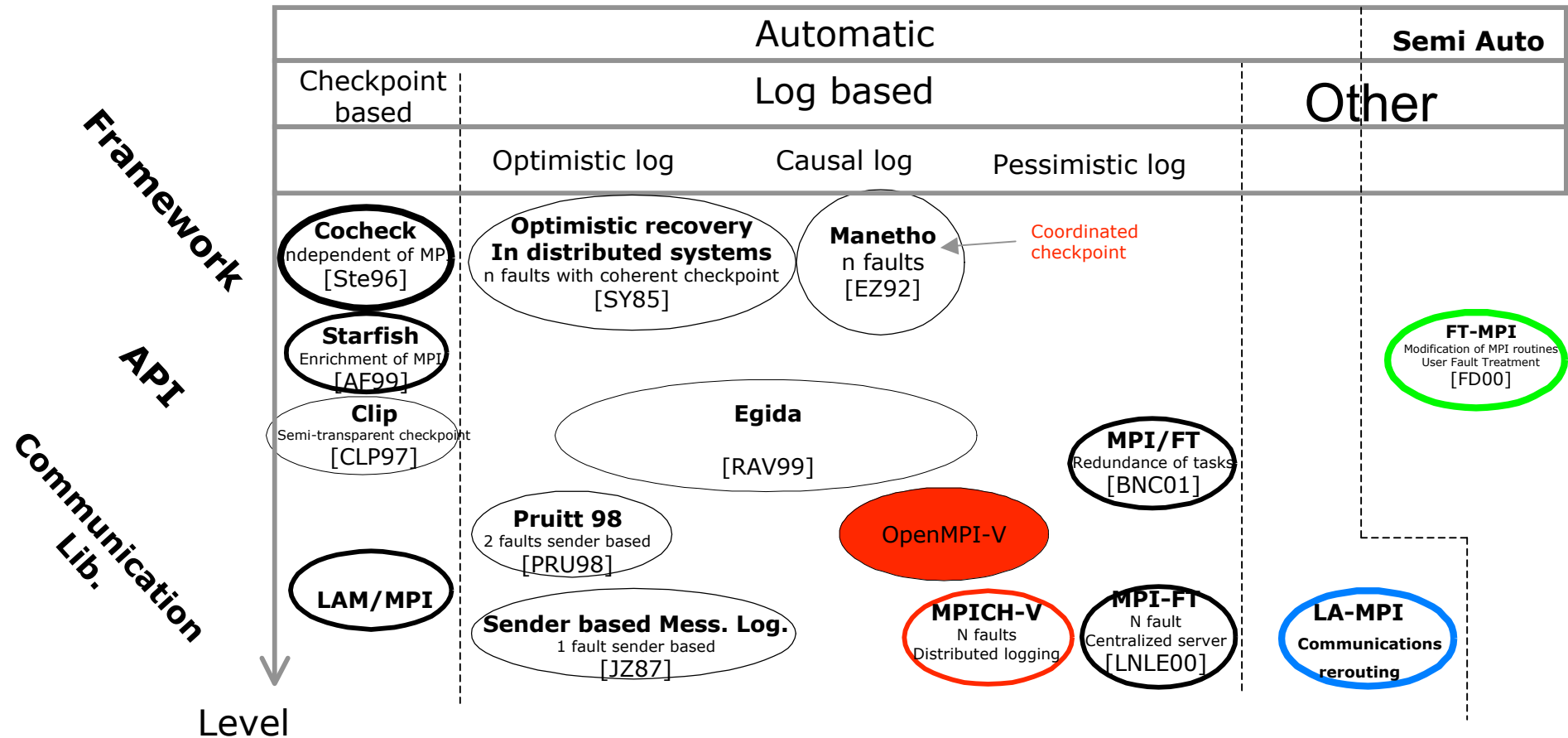
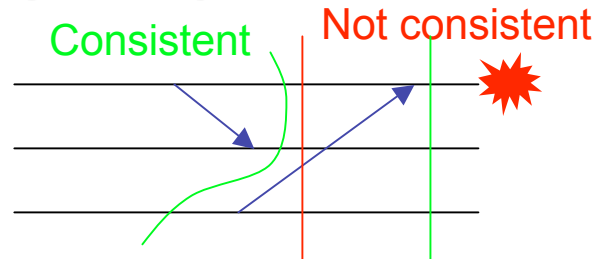
- Why Fault Tolerance is Challenging?
- What are the main reasons behind failures?
- What is the general trend about failure rates?
- Rollback recovery or Replication?
- **Rollback recovery Protocol?**
- Optimization of Rollback Recovery?
- Alternatives to Rollback recovery?
- Where are the opportunities?

Roll-Back Recovery Protocols: Which one to choose from?

P0
P1
P2



Where to checkpoint
to ensure a consistent
rollback-recovery?



Latest results in RR protocols

- Fully-Non-Blocking Coordinated ckpt. (comp. & comm. continue)
- Improved message logging
- Zoning

Ex: improved sender based message logging in MPI-V

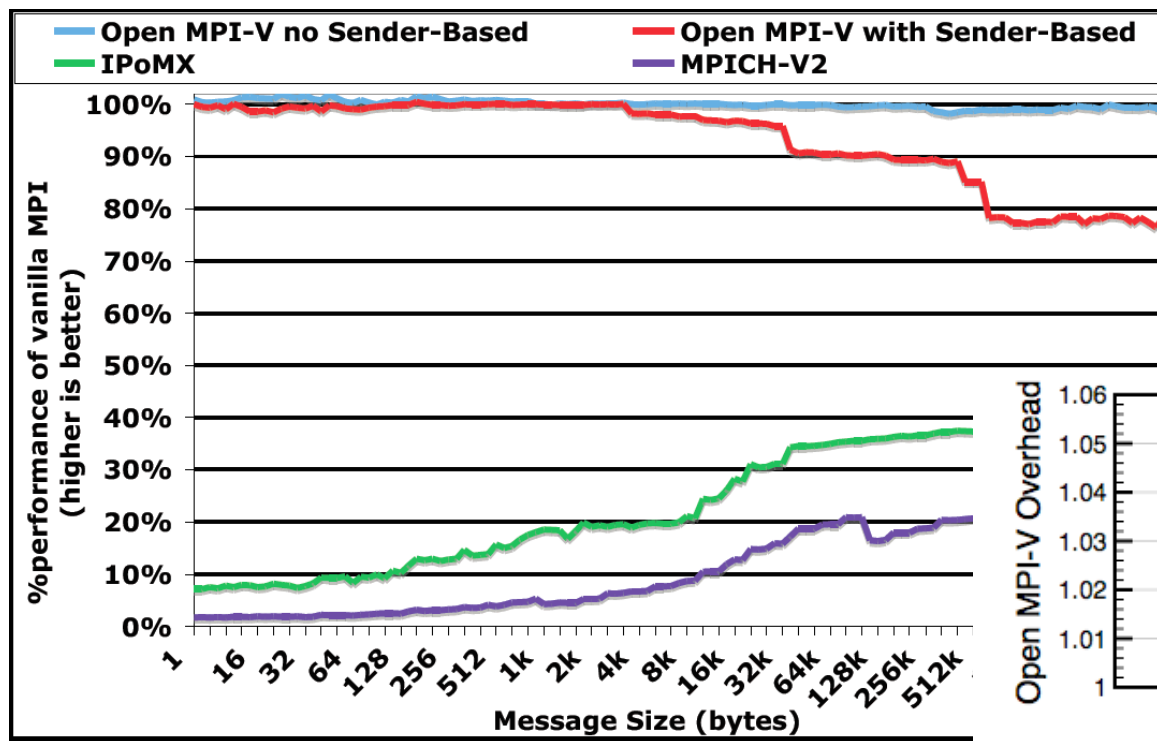
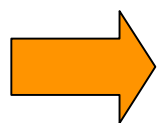
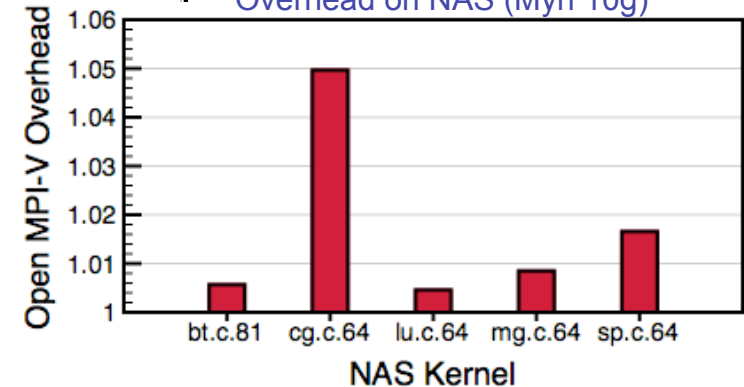


Fig. from
Boutieller

Overhead on NAS (Myri 10g)

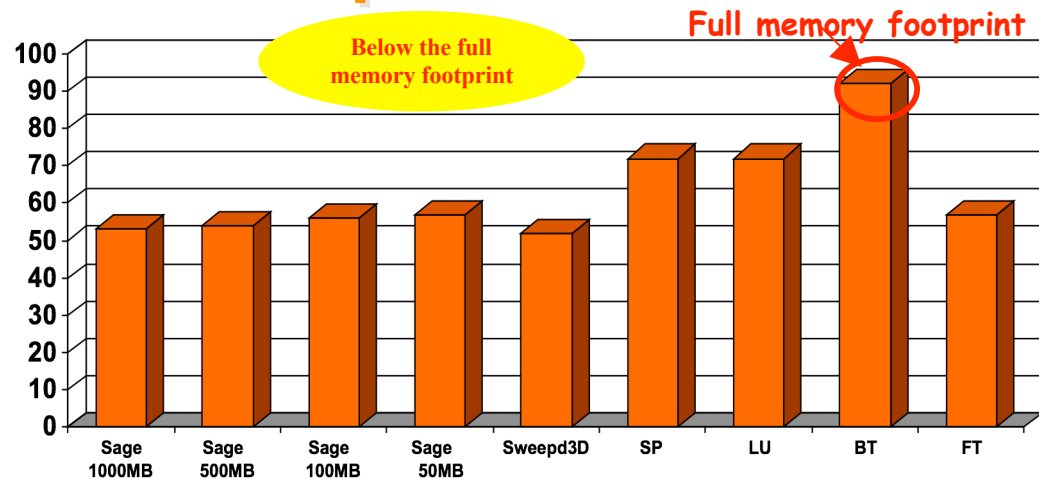


Coordinated and message logging protocols have been improved --> not much to gain in this domain

Reducing Checkpoint size

Fig. from J.-C. Sancho

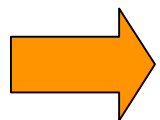
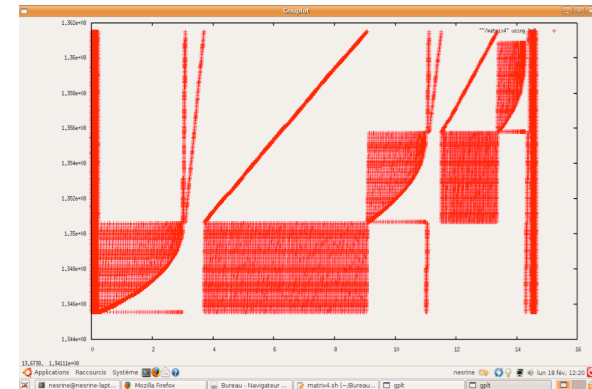
- System based incremental checkpointing



- Compiler assisted application level checkpoint (Cornel)
- Application checkpointing library (CEA)
- User specific codes
- Compression

- Inspector Executor (trace based) checkpoint (INRIA study)

Ex: DGETRF (max gain 20% over IC)

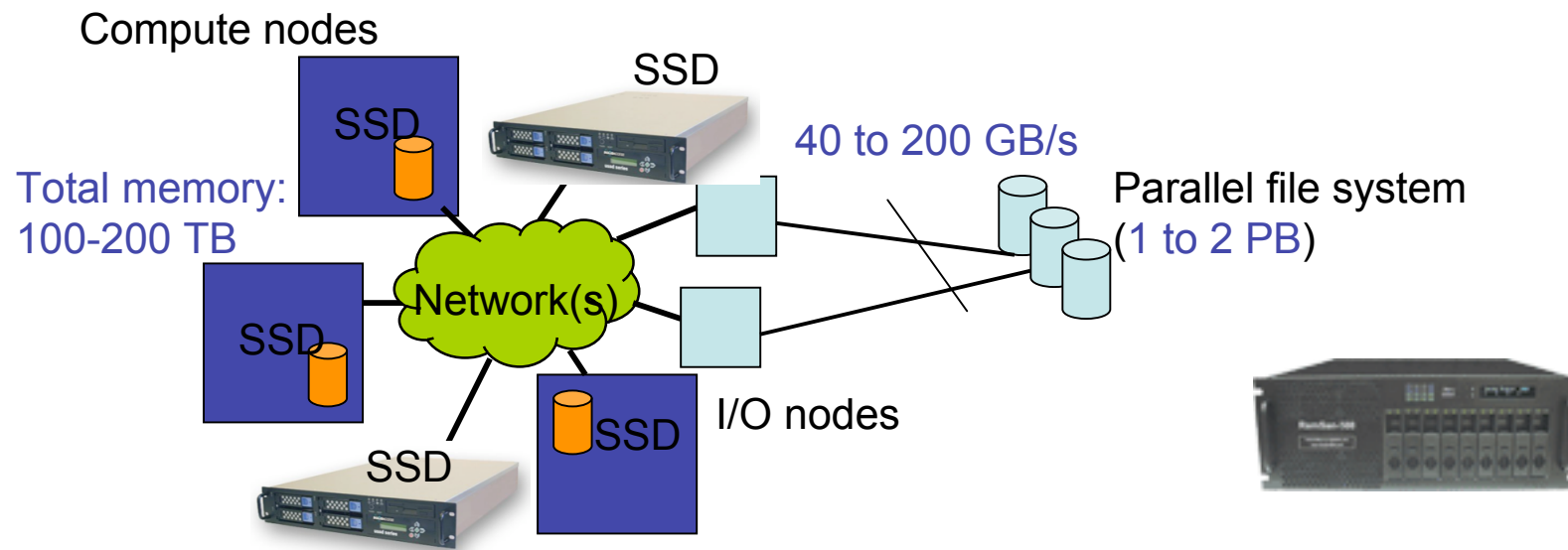


Challenge: Reducing checkpoint size is probably the most difficult problem but lead to strong returns

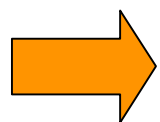
Some Issues related to fault tolerance of parallel applications

- Why Fault Tolerance is Challenging?
- What are the main reasons behind failures?
- What is the general trend about failure rates?
- Rollback recovery or Replication?
- Rollback recovery Protocol?
- Optimization of Rollback Recovery?**
- Alternatives to Rollback recovery?
- Where are the opportunities?

Store ckpt on SSD (Flash mem.)



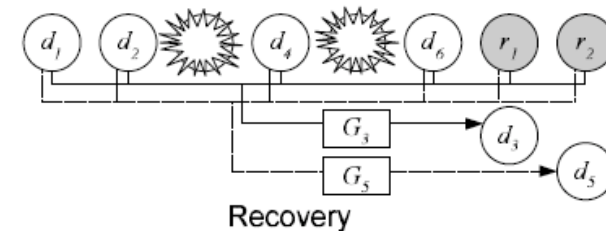
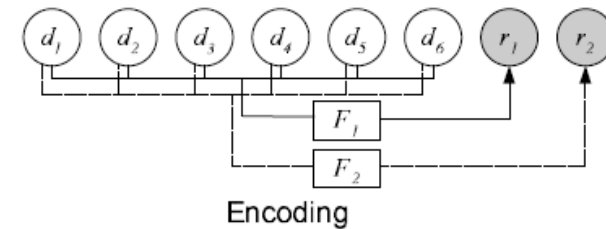
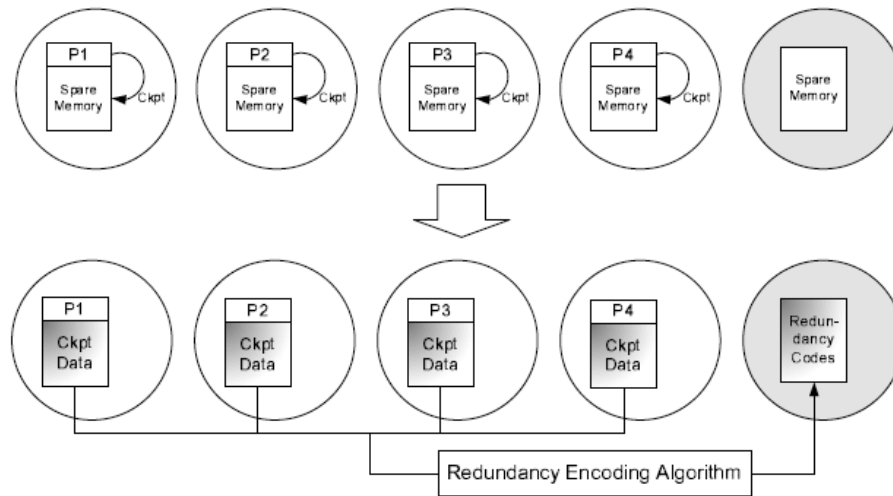
- Current practice --> checkpoint on local disk (1 min.) and then move asynchronously ckpt images to persistent storage
--> Checkpoint still needs 20-40 minutes (does not solve the 30min. Wall!)
- Recent proposal: flash memory (SSD) in nodes or attached to network
--> Increase the cost of the machine (100 TB or flash memory) + increase power consumption + need to store more than 1 ckpt images on remote nodes (if SSD on nodes) OR add a large # of components in the system.



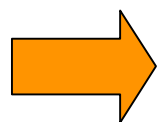
Challenge: How to integrate the SSD tech. while keeping reasonable cost and #failures?

Diskless Checkpointing

Images from
CHARNG-DA LU



- Need spare nodes --> increases the overall cost and #failures
- Tolerate only N simultaneous faults (depends on coding)
- Need a global synchronization & freeze during ckpt.
- Need very fast encoding & reduction operations
- Requires a automatic system level Ckpt protocol or program modifications
- Work with with incremental ckpt.



Challenge: Remove the need for synchronization and global Rollback

Some Issues related to fault tolerance of parallel applications

- Why Fault Tolerance is Challenging?
- What are the main reasons behind failures?
- What is the general trend about failure rates?
- Rollback recovery or Replication?
- Rollback recovery Protocol?
- Optimization of Rollback Recovery?
- Alternatives to Rollback recovery?
- Where are the opportunities?

Wrapping-up

Fault tolerance should be considered as a major issue and integrated early in the design of a PetaScale system

Coordinated and message logging protocols have been improved --> not much to gain in this domain

Challenges:

- Reduce the cost of Checkpointing (checkpoint size & time)
- Design better logging and analyzing tools
- Design less expensive replication approaches
- Integrate Flash mem. tech. & keep low cost and #failures?
- Remove global synch. and rollback in Diskless ckpt.

Opportunities: we can still try to optimize...

Table from
Bouteiller

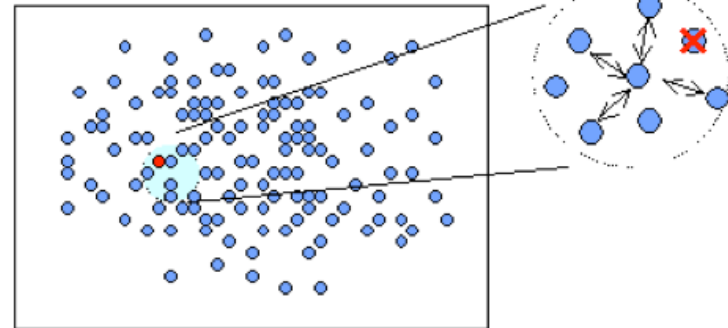
	BT	SP	FT	CG	MG						LU					
#processors	all				4	32	64	256	512	1024	4	32	64	256	512	1024
%non-deterministic	0	0	0	0	40.33	29.35	27.10	22.23	20.67	19.99	1.13	0.66	0.80	0.80	0.75	0.57

- Remove sources of non-determinism
- Checkpoint only at selected moments
- Limit fault types (ex: network --> MPI-FT - network)
- Restart at the latest checkpoint (ex: optimistic protocols -- subject to domino effect)
- Save all data “simultaneously” (relax global coordination)
--> needs message logging between ckpt regions.
- Specific FT approach for specific patterns (ex: most communication patterns are deterministic)

Opportunities: other FT paradigms?

- Almost all current researches are based on variations of the original Chandy-Lamport algorithm for “determination of consistent global states”
 - This algorithm does not consider all types of failures (Byzantine)
 - It is mostly used to restart the execution in the same conditions as before the failure
 - It concerns applications that cannot “survive” to failures

Figure from
A. Geist

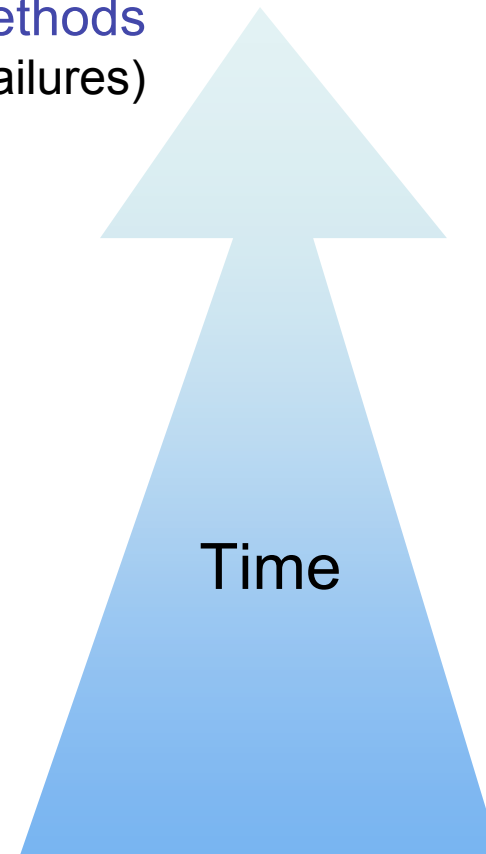


Meshless formulation of 2-D finite difference application

- “Naturally fault tolerant algorithm”
(Al Geist Christian Engelmann)
Ex: Meshless iterative methods
+chaotic relaxation
- Other distributed system paradigms considering failures are “normal” events:, Speculative Multi-Threading, Software Transactional Memory, Self-Stabilization

Conclusion: need for short and long term researches

- Novel computing methods
(inherently tolerant to failures)
- Novel FT paradigm
 - STM, SelfSta, Speculation?
 - Specific hardware?
- Optimizations:
 - Checkpoint Storage & transfers
(Additional Local disk, Flash mem.)
 - Reduce checkpoint size
 - Proactive Migration (sensors)
 - Replication?



Definitively, we would be interested in collaborating with Japan top 4 ;-)< --> what about FT and Language?